



# Binary Sight-Seeing: Accelerating Reverse Engineering via Point-of-Interest-Beacons

August See, Maximilian Gehring,  
Mathias Fischer, Shankar Karuppayah



# Motivation

## ■ Reverse Engineering

- Time consuming
- Driven by experience and beacons
  - E.g., function names and strings

Identify points of interest (POIs) within binary code

## ■ Goal: Provide More Useful Beacons

- Instructions that interact with specific data
- Example: Where is the message of type Oxcafe constructed?

Filter the POIs on their relevance

Evaluation on ransomware and P2P botnets

Automated botnet monitoring (peer list extraction and crawling)

## Core Idea for Finding POI

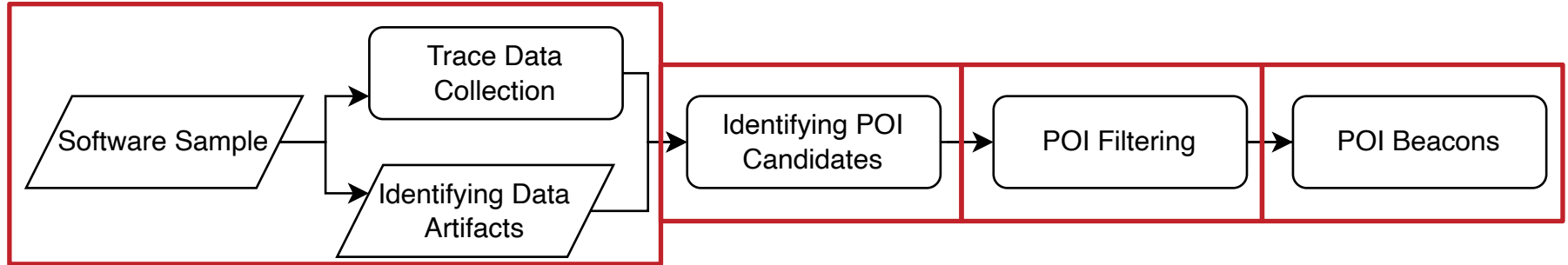
- Identification of Points of Interest (POI)
  - Locations in a binary which the analyst is interested in
- POIs are Dependent on Data
  - "Points where the binary interacts with data X"
    - Encryption function ⇒ Location where cleartext and ciphertext content is accessed
  - Data is often observable or known
    - Traffic, file content, ...

---

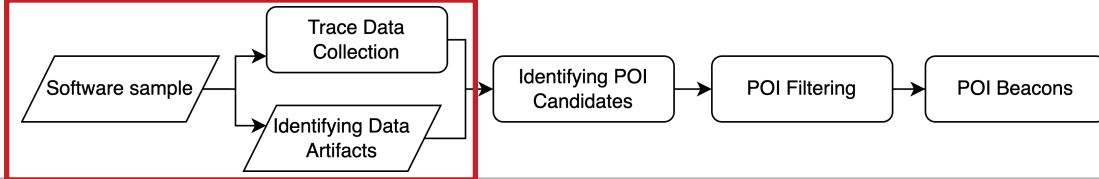
```
1 // eax, address of the message buffer
2 mov [eax], 0xcafe
3 // further message construction
4 push eax
5 // send message buffer to recipient
6 call send_message
```

---

## Overview of Using our Approach on a Binary



# Preparation



## ■ Identifying Data Artifacts via Monitoring Software

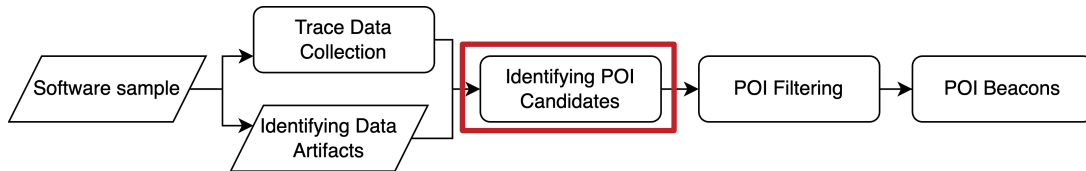
- Any.run, Cuckoo, Wireshark, Procmon, ...
- Or prior knowledge

1804	12.288001	192.168.178.140	192.168.178.255	UDP	63	35205 → 32414	Len=21
1805	12.288003	192.168.178.140	192.168.178.255	UDP	63	53241 → 32412	Len=21
1806	12.297125	192.168.178.129	185.70.42.43	TLSv1...	311	Application Data	
1807	12.299523	192.168.178.129	185.70.42.55	TCP	78	62023 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3309453112 TSecr=0 SACK_PERM	
1808	12.321176	185.70.42.43	192.168.178.129	TCP	66	443 → 61287 [ACK] Seq=13737 Ack=441 Win=1022 Len=0 TSval=2402301247 TSecr=3132671460	
1809	12.401501	185.70.42.43	192.168.178.129	TCP	1294	443 → 61287 [ACK] Seq=13737 Ack=441 Win=1022 Len=1228 TSval=2402301328 TSecr=3132671460 [T	
1810	12.401504	185.70.42.43	192.168.178.129	TLSv1...	322	Application Data	
1811	12.401781	192.168.178.129	185.70.42.43	TCP	66	61287 → 443 [ACK] Seq=441 Ack=15221 Win=2024 Len=0 TSval=3132671564 TSecr=2402301328	
1812	12.697849	192.168.178.70	255.255.255.255	UDP	214	59727 → 6667	Len=172

## ■ Trace Data Collection

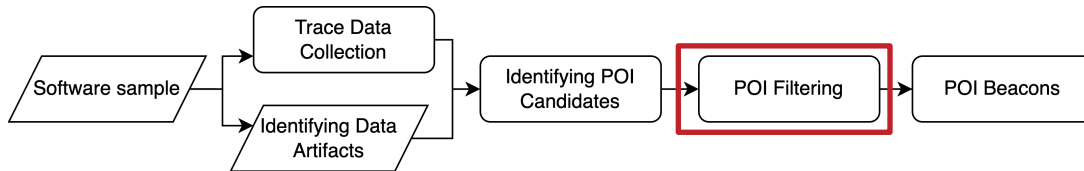
- Run our tool to get the execution trace of the program
- Trigger wanted behavior

# Identifying POI Candidates



- Determine whether an instruction interacted with the data artifacts
- Single Instruction POIs: Complete data artifact is accessed in one instruction
  - Check accessed memory and registers
    - **mov** eax, 0xcafe [Register POI]
    - **push** eax [Memory POI]
- Multi Instruction POIs
  - Complicated as data might be accessed in random order
    - Hello World
  - Solution: Search surrounding memory area when a searched byte is written
    - Optimize by checking memory allocations

# Filtering POIs



## ■ Found POIs might be noisy

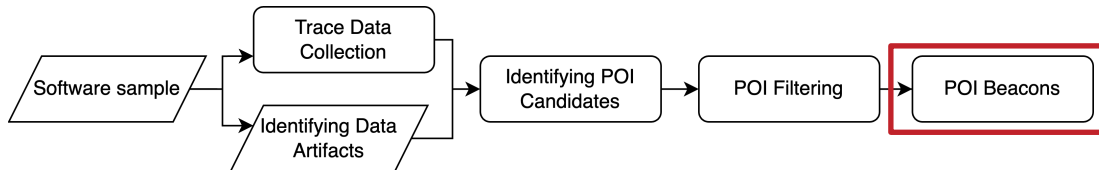
- Data propagates from function to function
- Not every function is of interest
  - `readfile` → `memcpy` → `encrypt` → `memcpy` → `writefile`

## ■ New Metric: Confidence Score

- Calculate how exclusive a function operates on our data
- $D :=$  set of data artifacts for identifying POIs
- Single Instruction POI
  - Determine the ratio of accessed data in  $D$  to the total accessed data
- Multi Instruction POI
  - Calculate the ratio of accessed bytes in  $D$  to the total accessed bytes
- The more complete  $D$  is, the higher the scores

$$score_S(p) := \sum_{d \in D} \frac{C(d, \text{trace}(p))}{|\text{trace}(p)|}$$

# POI Beacons Plugin for IDA



POI := Address, confidence score, list of accessed patterns

IDA View-A

```

mov     eax, [esp+34h+arg_4]
sar     ebx, 10h
mov     [eax], dl ; Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor
xor     edx, edx
mov     dl, byte ptr [esp+34h+var_20+2]
mov     [esp+34h+arg_0], edi
mov     dl, ds:byte_10007A3C[edx]
xor     dl, bl
mov     ebx, edi
mov     [eax+1], dl ; Encrypted-File-Content | Rank 2 | w | CLIContiguousPoiExtractor
xor     edx, edx
mov     dl, ch
sar     ebx, 8
mov     dl, ds:byte_10007A3C[edx]
          
```

Pseudocode-G

```


91  *a3 = HIBYTE(v20) ^ byte_10007A3C[HIBYTE(v25)];
92  a3[1] = BYTE2(v20) ^ byte_10007A3C[BYTE2(v24)];
93  a3[2] = BYTE1(v20) ^ byte_10007A3C[BYTE1(v7)];
94  a3[3] = v20 ^ byte_10007A3C[(unsigned __int8)v23];
95  v31 = v21[1];
96  a3[4] = HIBYTE(v31) ^ byte_10007A3C[HIBYTE(v24)];
97  a3[5] = ((unsigned __int16)(v31 >> 8) >> 8) ^ byte_10007A3C[(unsigned __int8)v26];
98  a3[6] = BYTE1(v31) ^ byte_10007A3C[BYTE1(v23)];
99  a3[7] = v31 ^ byte_10007A3C[(unsigned __int8)v25];
100 v32 = v21[2];
101 a3[8] = HIBYTE(v32) ^ byte_10007A3C[HIBYTE(v26)];
102 a3[9] = ((unsigned __int16)(v32 >> 8) >> 8) ^ byte_10007A3C[BYTE2(v23)];
103 a3[10] = BYTE1(v32) ^ byte_10007A3C[BYTE1(v25)];
104 a3[11] = v32 ^ byte_10007A3C[(unsigned __int8)v24];
          
```

Type	Location	Pass count	Hardware	Condition	Actions	State	Comment	Group
Abs	0x100069DA (sub_10006940:loc_100069DA)				Break	Disabled	Cleartext-File-Content   Rank 1   r   CLIContiguousPoiExtractor	Default
Abs	0x100069DD (sub_10006940+9D)				Break	Disabled	Encrypted-File-Content   Rank 1   r   CLIContiguousPoiExtractor	Default
Abs	0x10006A0D (sub_10006940+CD)				Break	Disabled	Encrypted-File-Content   Rank 1   r/w   CLIContiguousPoiExtractor	Default
Abs	0x100064CC (sub_10006280+24C)				Break	Disabled	Encrypted-File-Content   Rank 2   w   CLIContiguousPoiExtractor	Default
Abs	0x100064E2 (sub_10006280+262)				Break	Disabled	Encrypted-File-Content   Rank 2   w   CLIContiguousPoiExtractor	Default
Abs	0x100064F8 (sub_10006280+278)				Break	Disabled	Encrypted-File-Content   Rank 2   w   CLIContiguousPoiExtractor	Default
Abs	0x1000650D (sub_10006280+28D)				Break	Disabled	Encrypted-File-Content   Rank 2   w   CLIContiguousPoiExtractor	Default




## Evaluation – Research Questions

---

- RQ1: Finding instructions that process the relevant data are helpful for RE
  - Shown selectively for Ransomware and P2P botnets 
- Core idea: Can the approach find functions a reverse engineer is interested in
  - Encryption function of the ransomware
  - P2P management function of the botnet

## Evaluation – Research Questions

---

- RQ2: The more exclusive a POI interacts with relevant data the better it is
  - Confidence score predicts quality
- Objective: Find the instruction that interacts with the peer list 
  - Then we can extract peers by monitoring this instruction
  - Input Data: IPs obtained by monitoring network connections ( $P_{\text{bootstrap}}$ )
  - Nice: We have (partial) ground truth, i.e., total knowledge of the peers

# Evaluation – Botnet Background

---

- P2P Botnets
  - No central C&C structure but peer lists
    - $P_{\text{bootstrap}} :=$  Initial peer list
  - Exchange of peers in set intervals
  
- Botnet Sample Preparation
  - Obtain Initial peer list
    - Monitor contacted IPs
  - Determine crawling interval
    - Time till IP is contacted again

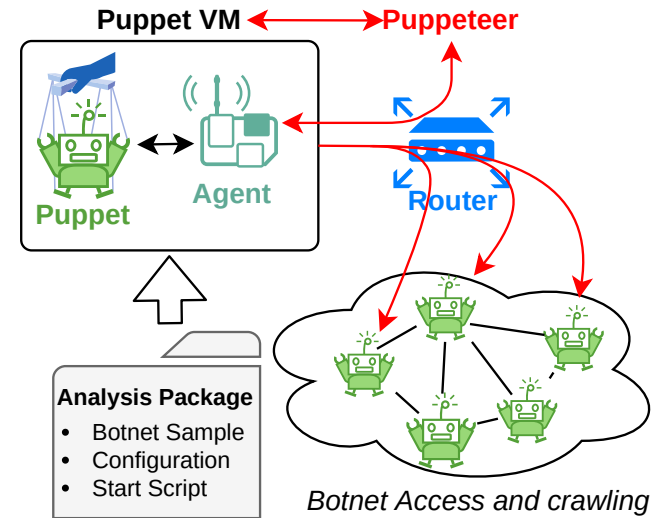
# Evaluation – Setup

## Local P2P Botnet

- Infect 40 VMs with the botnet
- Let them find each other in the local network
  - $P_{local} :=$  Available local peers
  - Peer list of a bot should only contain  $P_{bootstrap} \cup P_{local}$ 
    - (Partial) Ground truth
- Observed botnets: Sality, Nugache, ZeroAccess, Kelihos

## VMs are orchestrated via Proxmox

- Snapshot and rollback functionality allows us
  - to determine crawling interval and  $P_{bootstrap}$
  - to repeatedly send peer list messages
- Router redirects P2P messages, e.g., get peer list



## Evaluation – Found POIs

### ■ Use each found POI to extract new peers ( $P_{local}$ )

#### – Peer can be

- from  $P_{bootstrap}$
- from  $P_{local}$  (correct peers)
- not from either → wrong

### ■ For each Botnet multiple POIs could be found

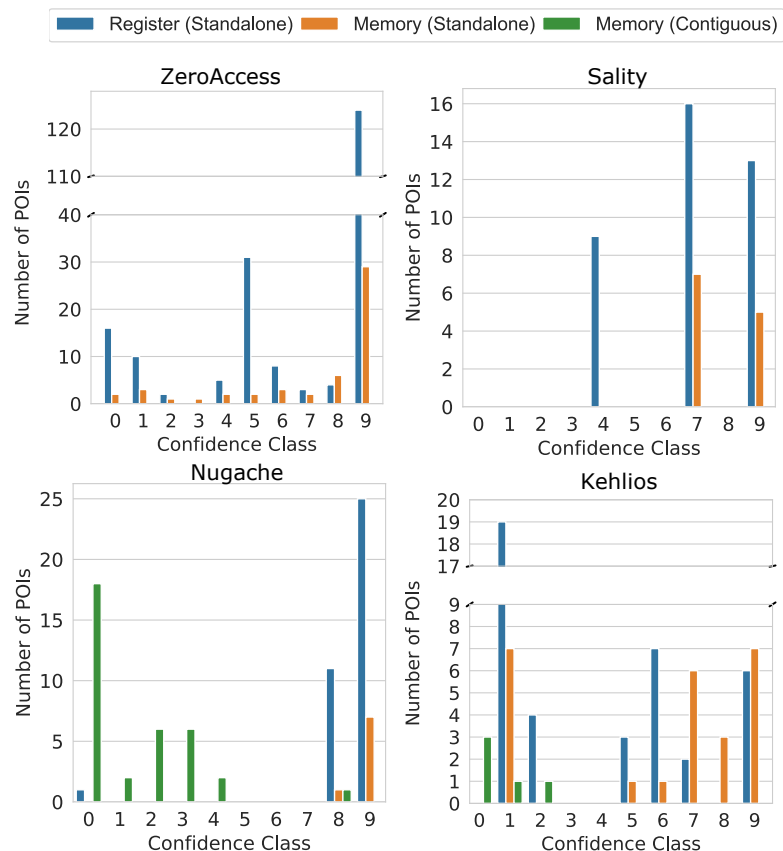
#### – Different instruction address

### ■ High Confidence Score for Standalone POIs

#### – IPs can fit in 4 Bytes

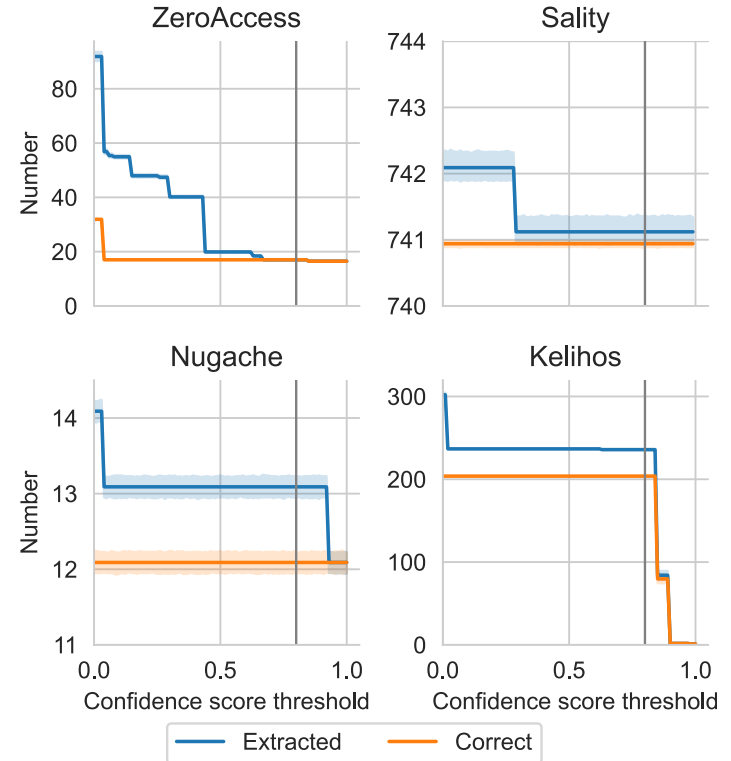
### ■ Low confidence score POIs are low quality?

#### – Cannot extract valid peers?



## Evaluation – Confidence Score

- Use each found POI to extract new peers ( $P_{local}$ )
  - from  $P_{bootstrap}$
  - from  $P_{local}$  (correct peers)
  - not from either  $\rightarrow$  wrong
- Cumulative Plot
  - We plot for confidence  $x$  the number of POIs for which the confidence score is  $\geq x$
  - Average of extracting peers for one membership cycle
- Amount of wrong peers decreases
  - Confidence score predicts quality of POIs



## Evaluation – Confidence Score

		Without confidence score threshold		With confidence score threshold ( $\geq 0.8$ )	
		$\emptyset$ ( $n = 100$ )	Avg. confidence score	$\emptyset$ ( $n = 100$ )	Avg. confidence score
ZeroAccess	$ P_{\text{extracted}} $	91.89 ( $\sigma \approx 8.62$ )	0.44 ( $n = 100$ $\sigma \approx 0.01$ )	17.00 ( $\sigma \approx 0.14$ )	0.94 ( $n = 100$ $\sigma \approx 0.01$ )
	CORRECT	15.91 ( $\sigma \approx 0.51$ )	0.48 ( $n = 100$ $\sigma \approx 0.05$ )	1.00 ( $\sigma \approx 0.14$ )	0.85 ( $n = 99$ $\sigma \approx 0.01$ )
	BOOTSTRAP	16.00 ( $\sigma \approx 0.00$ )	0.62 ( $n = 100$ $\sigma \approx 0.01$ )	16.00 ( $\sigma \approx 0.00$ )	0.96 ( $n = 100$ $\sigma \approx 0.01$ )
	WRONG	59.98 ( $\sigma \approx 8.39$ )	0.20 ( $n = 100$ $\sigma \approx 0.01$ )	0.00 ( $\sigma \approx 0.00$ )	n/a
Salaty	$ P_{\text{extracted}} $	742.09 ( $\sigma \approx 1.11$ )	1.00 ( $n = 100$ $\sigma \approx 0.00$ )	741.12 ( $\sigma \approx 1.07$ )	1.00 ( $n = 100$ $\sigma \approx 0.00$ )
	CORRECT	1.94 ( $\sigma \approx 0.24$ )	0.83 ( $n = 100$ $\sigma \approx 0.03$ )	1.94 ( $\sigma \approx 0.24$ )	1.00 ( $n = 100$ $\sigma \approx 0.00$ )
	BOOTSTRAP	739.00 ( $\sigma \approx 0.00$ )	1.00 ( $n = 100$ $\sigma \approx 0.00$ )	739.00 ( $\sigma \approx 0.00$ )	1.00 ( $n = 100$ $\sigma \approx 0.00$ )
	WRONG	1.15 ( $\sigma \approx 1.05$ )	0.29 ( $n = 97$ $\sigma \approx 0.02$ )	0.18 ( $\sigma \approx 1.03$ )	1.00 ( $n = 3$ $\sigma \approx 0.00$ )
Nugache	$ P_{\text{extracted}} $	14.09 ( $\sigma \approx 0.72$ )	0.98 ( $n = 100$ $\sigma \approx 0.00$ )	13.09 ( $\sigma \approx 0.72$ )	0.99 ( $n = 100$ $\sigma \approx 0.00$ )
	CORRECT	3.45 ( $\sigma \approx 0.80$ )	0.98 ( $n = 100$ $\sigma \approx 0.01$ )	3.45 ( $\sigma \approx 0.80$ )	0.99 ( $n = 100$ $\sigma \approx 0.00$ )
	BOOTSTRAP	8.64 ( $\sigma \approx 0.71$ )	0.99 ( $n = 100$ $\sigma \approx 0.00$ )	8.64 ( $\sigma \approx 0.71$ )	0.99 ( $n = 100$ $\sigma \approx 0.00$ )
	WRONG	2.00 ( $\sigma \approx 0.00$ )	0.48 ( $n = 100$ $\sigma \approx 0.00$ )	1.00 ( $\sigma \approx 0.00$ )	0.92 ( $n = 100$ $\sigma \approx 0.00$ )
Kelihos	$ P_{\text{extracted}} $	301.98 ( $\sigma \approx 12.89$ )	0.63 ( $n = 100$ $\sigma \approx 0.06$ )	235.79 ( $\sigma \approx 1.49$ )	0.85 ( $n = 100$ $\sigma \approx 0.00$ )
	CORRECT	40.79 ( $\sigma \approx 1.49$ )	0.46 ( $n = 100$ $\sigma \approx 0.12$ )	40.79 ( $\sigma \approx 1.49$ )	0.85 ( $n = 100$ $\sigma \approx 0.00$ )
	BOOTSTRAP	163.00 ( $\sigma \approx 0.00$ )	0.76 ( $n = 100$ $\sigma \approx 0.03$ )	163.00 ( $\sigma \approx 0.00$ )	0.85 ( $n = 100$ $\sigma \approx 0.00$ )
	WRONG	98.19 ( $\sigma \approx 12.41$ )	0.72 ( $n = 100$ $\sigma \approx 0.02$ )	32.00 ( $\sigma \approx 0.00$ )	0.85 ( $n = 100$ $\sigma \approx 0.00$ )

Partial Ground Truth

## Discussion

---

- **Advantages of Using Local Botnets for Evaluation**
  - Utilization of Third-Party Malware Samples: Ensures real-world testing scenarios
  - Access to Ground Truth Data: Facilitates of results
  - High Observability: Allows detailed monitoring and analysis of botnet behavior
- **Acceleration of Reverse Engineering**
  - Evidential Gap: How do we effectively demonstrate or measure the acceleration of reverse engineering processes?
- **Defining and Comparing Usability**
  - What are our benchmarks or comparatives for assessing usability enhancements?
- **What would a user study look like that would evaluate the speed and or usability**
  - What other evaluation methods might be feasible?



## Additional Results and Failures

---

### ■ Online Monitoring and Modification

- Enables redirection of destination dependent payloads
  - Change destination IP as soon as it is in memory
  - Depending on the botnet's implementation
- Problem: Timeouts due to runtime of our instrumentation and analysis

### ■ Automatic Key Extraction

- Collect and merge multiple instruction traces
  - Hope that key depended instruction remain
- Differential Debugging

## Future Work

---

- User Study to Evaluate Acceleration and Useability
- Support for more complicated Pattern matching
  - E.g., regex: 192.168.XXX.101
- Efficient Handling of Online Monitoring
  - Increasing Performance
- Replacement of Intel PIN?
  - Multi architecture and OS capability

Thanks for your attention!

---

Questions?  
Thoughts?  
Ideas?

August See  
Universität Hamburg  
Computer Networks (NET)  
[richard.august.see@uni-hamburg.de](mailto:richard.august.see@uni-hamburg.de)